

jc639 U.S. PRO
05/07/99

Patent
Attorney's Docket No. P2380-505

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

UTILITY PATENT
APPLICATION TRANSMITTAL LETTER

JC542 U.S. PRO
09/306888
05/07/99

Box PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Enclosed for filing is the utility patent application of David G. Opstad and Alexander B. Beaman for AUTOMATIC SYNTHESIS OF FONT TABLES FOR CHARACTER LAYOUT.

Also enclosed are:

- ☒ 5 sheet(s) of ☒ formal ☐ informal drawing(s);
- ☐ a claim for foreign priority under 35 U.S.C. §§ 119 and/or 365 is ☐ hereby made to __
filed in __ on ____;
☐ in the declaration;
- ☐ a certified copy of the priority document;
- ☐ a Constructive Petition for Extensions of Time;
- ☐ statement(s) claiming small entity status;
- ☒ an Assignment document;
- ☐ an Information Disclosure Statement; and
- ☐ Other:

The declaration of the inventor(s) ☒ also is enclosed ☐ will follow.

- ☐ Please amend the specification by inserting before the first line the sentence --This application claims priority under 35 U.S.C. §§119 and/or 365 to __ filed in __ on __; the entire content of which is hereby incorporated by reference.--

The filing fee has been calculated as follows [] and in accordance with the enclosed preliminary amendment:

C L A I M S

	NO. OF CLAIMS		EXTRA CLAIMS	RATE	FEE
Basic Application Fee					\$760.00
Total Claims	31	MINUS 20 =	11	x \$18.00	198.00
Independent Claims	6	MINUS 3 =	3	x \$78.00	234.00
If multiple dependent claims are presented, add \$260.00					-0-
Total Application Fee					1192.00
If verified Statement claiming small entity status is enclosed, subtract 50% of Total Application Fee					-0-
Add Assignment Recording Fee of \$40.00 if Assignment document is enclosed					40.00
TOTAL APPLICATION FEE DUE					1232.00

- [] A check in the amount of \$ is enclosed for the fee due.
[X] Charge \$ 1232.00 to Deposit Account No. 02-4800 for the fee due.

Please address all correspondence concerning the present application to:

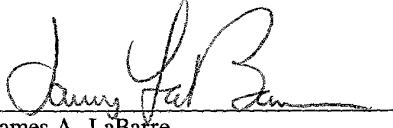
James W. Peterson
Burns, Doane, Swecker & Mathis, L.L.P.
P.O. Box 1404
Alexandria, Virginia 22313-1404.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§ 1.16, 1.17 and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800. This paper is submitted in triplicate.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

Date: May 7, 1999

By: 
James A. LaBarre
Registration No. 28,632

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

JC542 U.S. PTO
09/306888
05/07/99

**APPLICATION FOR UNITED STATES
LETTERS PATENT**

by

DAVID G. OPSTAD

and

ALEXANDER B. BEAMAN

for

**AUTOMATIC SYNTHESIS OF FONT
TABLES FOR CHARACTER LAYOUT**

BURNS, DOANE, SWECKER & MATHIS, LLP
P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

Attorney Docket: P2380:505

AUTOMATIC SYNTHESIS OF FONT TABLES FOR CHARACTER LAYOUT

Field of the Invention

5 The present invention relates to the display and printing of font characters, and more particularly to a method and system for automatically synthesizing font tables that determine the manner in which images of characters are processed for printing and display.

Background of the Invention

10 The continued evolution of application programs, such as word processing and desktop publishing programs, has provided users with a great deal of versatility and control in the appearance of documents created with these types of programs. A significant factor in this area is the ability to influence the layout of textual and symbolic characters on a page. Initially, the character fonts that were available for use in these types of programs were relatively static, in the sense that
15 their shapes and positioning were rigidly fixed according to a predefined set of rules. As new fonts have been developed, the technology pertaining to the appearance and layout of characters has also evolved, so that new capabilities are continuously being added to fonts. As a result, it is now possible to control a number of different parameters which determine the appearance and layout of
20 characters in a line of text. Examples of these parameters include the identification of the specific glyphs that define the appearances of the characters, substitution of contextual letterforms and ligatures, the positioning of the glyphs relative to one another, hanging punctuation, and optical alignment of a line of characters.

25 To provide these types of capabilities, a given font is comprised of more than just the glyphs which determine the appearances of the individual characters. In particular, the font definition consists of a number of data tables that relate to various parameters which control the implementation of the font. For instance, a TrueType font comprises a file that may contain up to thirty or more tables that determine the layout and other characteristics of the font. Examples of the types
30 of data in these tables include kerning, i.e. spacing between characters, the metrics

or dimensions of glyphs, variable properties such as line widths, and the like. The information provided by these tables is employed for a number of different purposes. For instance, some processes may rely upon a table of the names of various font styles, to display in a menu or dialog box. Another use of the tables occurs when a line of characters is to be laid out for display or printing purposes. One example of a technology which lays out a line of text pursuant to the data in such tables is described in U.S. Patent No. 5,416,898, the disclosure of which is incorporated herein by reference.

As font technology continues to evolve over time, new capabilities are constantly being added, to enhance the versatility and control over the appearance of characters. The added capabilities are generally accompanied by new data tables. An unfortunate consequence of this development is the fact that older fonts may not contain all of the tables necessary to conform to the latest capabilities. As an example, if an attempt is made to employ an older font which does not contain the latest tables, during the operation of a line layout processor that makes use of the newer tables, the results can range from an unaesthetic appearance to linguistically incorrect text.

In an effort to overcome the limitations associated with the use of older fonts, various tools have been made available to permit font developers to add new tables to existing fonts. However, there may be a reluctance by developers to alter the contents of a font once it has been accepted. Furthermore, even if a font is updated to incorporate the newer data tables, users must become aware of, and acquire, the updated fonts before they can be successfully used with the newer font processing technologies.

Accordingly, it is desirable to provide a mechanism via which data tables that are needed to properly process the glyphs of a font can be automatically synthesized if they are not part of the original font definition. Further in this regard, it is desirable to be able to synthesize and employ such tables in a manner which does not affect the original definition of the font.

Summary of the Invention

In accordance with the present invention, these objectives are achieved by means of a method and system in which one or more font tables are automatically synthesized as needed, and placed in an annex file that is associated with the original font. When a character or string of text is requested by an application program, for example, a computer's font system first checks the original definition of the font, to see if it contains the tables that are necessary to display or print the requested characters. If the font itself does not contain the tables, the font system examines an associated annex file to see if the appropriate tables exist. If so, the table is accessed and its data is employed to process font data. If the table does not exist, it is automatically synthesized and placed in the annex, where its data is then utilized.

Using an annex file that is associated with a font, rather than physically adding the tables to the fonts themselves, provides several advantages. In particular, the actual font data is not affected by the creation of the new table. Consequently, font management systems that employ font protection, for example by performing checksums or calculating digital signatures for font data, will continue to operate properly. Furthermore, since the annex file is persistent and survives across successive boots of the computer, the effort required to synthesize a table is only expended once per font, and thereafter available for all subsequent uses of the font.

In one embodiment, the synthesis of a table is carried out by first creating a font map. The font map is constructed by analyzing a glyph repertoire associated with the font, to identify certain types of data associated with each glyph. The data in the font map is then used to compute tables which identify specific types of mappings between glyphs, or other characteristics associated with individual glyphs. These tables are stored in the annex, and their data is employed for various font processing techniques. In another embodiment, a table is created by reformatting and/or translating data that is already present in a font, to place it in a structure that can be used by a particular process.

The foregoing features of the invention, and the advantages provided thereby, are explained in greater detail hereinafter with reference to specific embodiments illustrated in the accompanying drawings.

Brief Description of the Drawings

5 Figure 1 is a general block diagram of the components of an exemplary computer system;

 Figure 2 is a block diagram illustrating the general architecture of software components involved in the implementation of the present invention;

 Figure 3 is a more detailed view of the architecture of the font subsystem;

10 Figure 4 is a flow chart of the general procedures that are carried out in a line layout processor;

 Figure 5 is a schematic illustration of the relationship of a font suitcase and an annex file;

 Figure 6 is a flow chart of the general procedure for obtaining and
15 synthesizing font data tables;

 Figure 7 is a schematic illustration of one example of the synthesis of a data table; and

 Figure 8 is a schematic illustration of another example of the synthesis of a data table.

20 Detailed Description

 To facilitate an understanding of the present invention, it is described hereinafter in the context of a specific implementation thereof. In particular, reference is occasionally made to features of the invention that are employed in connection with TrueType fonts, as well as line layout technology that is intended
25 to be used with this type of font. It will be appreciated, however, that the practical applications of the invention are not limited to this particular example. For instance, the following embodiments have been demonstrated to work with PostScript fonts as well. Likewise, the data tables that are synthesized need not be

only those which are used for line layout. The principles which underlie the invention can be utilized in connection with a number of different types of font technologies and font handling systems in which it is useful to employ font tables that determine the characteristics of a font.

5 The present invention is broadly directed to the generation of character images in a computer, for display on a display device, such as a monitor, and/or printing in a document. While the particular hardware components of a computer system do not form a part of the invention itself, they are briefly described herein to provide a thorough understanding of the manner in which the features of the
10 invention cooperate with the components of a computer system to produce desired results. Referring to Figure 1, a typical computer system includes a computer 10 having a variety of peripheral devices 12 connected thereto. The computer 10 includes a central processing unit 14 and associated memory. This memory generally includes a main working memory which is typically implemented in the
15 form of a random access memory 16, a static memory that can comprise a read only memory 18, and a permanent storage device, such as a magnetic or optical disk 20. The CPU 14 communicates with each of these forms of memory through an internal bus 22. The peripheral devices 12 include a data entry device such as a keyboard 24, and a pointing or cursor control device 26 such as a mouse,
20 trackball, pen or the like. A display device 28, such as a CRT monitor or a LCD screen, provides a visual display of the information that is being processed within the computer, for example the contents of a document or a computer generated image. A tangible copy of this information can be provided through a printer 30, or similar such device. Each of these external peripheral devices communicates
25 with the CPU 14 by means of one or more input/output ports 32 on the computer.

 As further background to the concepts which underlie the present invention, one process which makes use of data tables synthesized in accordance with the invention, namely the generation and laying out of images of characters and symbols, will first be described. The general architecture of software
30 programs that are loaded into the RAM 16 and executed on the computer is

illustrated in the block diagram of Figure 2. In a typical situation, the user interacts with one or more application programs 34, such as a word processing program, a desktop publishing program, or a graphics program. In operation, as the user types words via the keyboard 24, the application program issues requests to the computer's operating system 36 to have the characters corresponding to the keystrokes drawn on the display 28. Similarly, when the user enters a command to print a document, the application program issues requests to the operating system which cause the corresponding characters to be printed via the printer 30. For illustrative purposes, the following description of the operation of the present invention will be provided for the example in which characters are drawn on the screen of the display 28 in response to user-entered keystrokes. It will be appreciated, however, that similar operations are carried out in connection with the printing of characters in a document on the printer 30.

When a user types a character via the keyboard 24, an indication of that event is provided to the application program 34 by the computer's operating system 36. In response, the application program issues a call to the computer's imaging system 38, to draw the character corresponding to the keystroke at a particular location on the display. That call includes a character code that designates a particular letter or other item of text, and style information which contains an identification of the font for the corresponding character. The imaging system 38 is a component of the computer's operating system 36. In the case of the Macintosh operating system, for example, the imaging system could be QuickDraw or QuickDraw GX.

Upon receipt of the request for a character in a particular font, the imaging system accesses a glyph cache 40, which contains bitmap images of characters. If the requested character has been previously displayed in the designated style, its image will be stored in the glyph cache, and immediately provided to the imaging system. If, however, the requested character is not stored in the cache, a call is made to a font subsystem 42, to obtain the requested image. The call to the font

subsystem identifies a particular font object, a point size, a resolution and any possible variations.

The structure of the font subsystem is illustrated in Figure 3. Within the font subsystem 42, the management of requested font objects is handled by a font server 44. In general, when a request for a font is received from an application program, for instance via the imaging system 38, the font server is responsible for locating the font or, if it is not available, the best substitute for it. The font server also retrieves the fonts that have been requested, and supplies them to the imaging system. The font server may also be responsible for displaying a font selection dialog box from which the user can choose a desired font, and/or providing a list of available fonts to the application program, to be displayed in a font menu.

Each font that is stored in the computer, for example on the hard disk 20, resides in a file that is sometimes referred to as a suitcase. Figure 3 illustrates a number of suitcases 46 that respectively store fonts labeled 1 through n . Each suitcase contains all of the data tables and other associated information that forms the definition of a font.

Fonts can be classified according to different technologies. Two well-known and widely used font technologies are TrueType and Type 1. These types of fonts are known as outline fonts, since their typeface designs, or glyphs, are specified by vectors which define the outlines of their shapes. Other types of fonts might fall into a category that is referred to as bit-mapped, or screen fonts, in which each pixel of a glyph is defined. Each of these technologies has a different set of rules for processing font data to satisfy a particular request. For instance, the characters of a bit-mapped font may be stored on the computer for a particular point size, e.g., 12 point. If the user requests the characters to be displayed at a different size, the pixel data must be processed to rescale the images of the characters. This processing is carried out in a font scaler 48a. In general, the font scaler operates in accordance with a set of rules for interpreting and processing the stored font data so as to satisfy specific requests from the font server 44. Each different type of font technology has an associated scaler for processing the data of

fonts which conform to that technology. Thus, in the example of Figure 3, a TrueType scaler 48b and a Type 1 scaler 48c are contained within the font subsystem.

In operation, the application program 34 issues a request for a particular font object. The font object provides an identification of a glyph family, e.g. a particular character such as "lowercase a", as well as its desired line weight and stress. The request from the application program also identifies any style variations to be applied to the character, such as italic, bold, underline, superscript, and the like. Upon receiving the font object, the font server 44 determines the font technology with which that object is associated, and passes the request on to the appropriate scaler 48. The scaler retrieves the necessary data from the suitcase 46 associated with the identified font, and processes it to meet the parameters specified in the request. The scaler then returns a glyph image to the font server 44, which is provided to the imaging system 38 and stored in the glyph cache 40.

As the individual glyph images are received from the font subsystem 42, they are displayed as a line of characters by the imaging system 38. The imaging system may include additional technology for processing and manipulating the glyph images before they are displayed. For example, the imaging system may include a line layout processor, which adjusts the positions of individual glyphs relative to one another, and performs further modifications of the glyphs, to lay out a complete line of characters. An example of the general procedures that are carried out in the line layout processor is illustrated in Figure 4. The line layout processor receives an input string consisting of a sequence of character codes and associated style information. As a first step, the character codes are mapped to glyph codes specific to the font. This mapping is performed in accordance with a data table stored in the font suitcase. The layout processor examines the glyph codes relative to certain rules and conditions, and selectively performs three major types of operations on the glyphs. These three operations constitute metamorphosis, positioning and justification. After these three processes have

been applied to the glyphs in a line, an output string is produced which contains bit-mapped information for drawing the appropriate glyph images.

Metamorphosis encompasses a variety of procedures via which the initial set of glyphs identified by the input string are transformed into a different set of glyphs. Some of the transformations can be independent of the context within which they appear. For instance, when a glyph for the letter ‘f’ is followed by a glyph for the letter ‘i’ in a word, these two glyphs can be transformed into an ‘fi’ ligature, in which the two separate glyphs appear to be joined. Another example of such transformation is the automatic conversion of numbers into Roman numeral form. Other types of transformations are dependent upon the context in which they appear, and are usually determined by rules pertaining to a specific language. For instance, two letters may be transposed if they appear at the end of a word, rather than in the middle of a word.

After the identities of glyphs have been changed in the metamorphosis step, the line layout processor determines whether the positions of any of the glyphs should be adjusted, relative to one another. For instance, in the case of the word ‘To’, it may be desirable to shift the position of the glyph for the letter ‘o’ to the left, to avoid the appearance of a large space between the two letters. Similarly, it may be appropriate to adjust the vertical position of a hyphen, so that it is centered with respect to the two characters between which it appears.

After the general positioning of the individual glyphs takes place, the final process in the layout of the line is the justification of the entire line. In this process, the intercharacter spacing is adjusted, as appropriate, to provide the desired type of justification. In some alphabets, the justification may be provided by extending the letters themselves, rather than by adjusting the spacing. For instance, the Arabic alphabet uses extension bars, known as kashidas, to increase the geometric length of a word, while keeping all characters connected to one another. The justification process determines whether the use of kashidas and/or intercharacter spacing adjustment is required, and revises the glyph images accordingly.

Each of these major processes relies upon the data contained in one or more font tables to carry out the appropriate adjustments to a sequence of glyphs. For instance, during the metamorphosis procedure, the layout processor may refer to a ligature table and a metamorphosis table to identify the appropriate glyph identity changes to be carried out. In operation, the layout processor employs the font objects associated with the glyphs in the input string to retrieve the tables corresponding to those fonts. These tables are then examined as part of the procedure to determine the types of glyph identity changes that need to be made. Similarly, during the positioning procedure, the layout processor may refer to tables that contain data relating to kerning, baseline alignment, tracking and glyph properties. During the justification procedure, the layout processor employs a justification table. In each case, the font objects associated with the glyphs in a line are employed to select the appropriate tables from the various font suitcases 46.

After the glyphs in a line have been appropriately processed through the layout procedure, an output string of pixel values is produced, to control the display of the line of characters on the display device 28. For more detailed information regarding the line layout procedures, and the manner in which the font data tables are employed in such procedures, reference is made to the aforementioned U.S. Patent No. 5,416,898.

Turning now to the specifics of the present invention, it may be the case that one or more of the fonts that are employed in a document does not include all of the tables that are necessary for the layout processor to perform all of its operations on the glyphs. This may be particularly true in the case of older fonts that were created before the capabilities of the layout processor were developed. In addition to the layout processor, other operations that occur within the computer may require information from data tables in the font. For instance, to display a menu of font styles, an application program may need to access a table of names. If the table is not present, the menu cannot be displayed. In accordance with the present invention, tables that are required to process fonts, for example within the

layout processor or in any other component of a font handling system, are automatically created when they are missing from the original font definition.

In operation, the automatic synthesis of a font table is carried out when the layout processor, or another software component which utilizes font data tables, issues a call for a specific table for a particular font, and the font subsystem determines that the data table is not present in that font. When this situation occurs, the automatic synthesis of the data table can be carried out in one of two ways. In one approach, all of the tables that are missing from the identified font can be synthesized and stored in the computer's random access memory 16 for the remainder of the session during which the font is utilized. This mode of operation may be advantageous when the same information about a font is used to synthesize a number of different tables. In this approach, it is only necessary to retrieve and analyze this information once. However, by synthesizing all missing tables at once, there may be an unnecessary expenditure of processing time, if all of the tables are not subsequently utilized during that session.

In another embodiment of the invention, the tables are individually synthesized as needed. Thus, for example, if the line layout processor issues a call for a metamorphosis table, and the font server determines that such a table is not present in the identified font, the metamorphosis table is synthesized at that time and used in the metamorphosis procedure. Subsequently, as additional tables are called for, they are individually synthesized as needed. With such an approach, the processing effort to synthesize the tables is limited to that which is actually required at the time.

In another aspect of the invention, the determination whether a table should be synthesized can be controlled on the basis of the table itself. Some data tables can be considered to be required as part of the basic implementation of the font. One example of such a data table is the table which maps character codes to glyph codes. Without such a table, many types of font operations cannot be carried out. Other types of data tables can be categorized as optional. For instance, some of the data tables employed by the layout processor may be language dependent, such

as a table which defines kashidas that are used in Arabic text but which are not employed with Roman characters. The determination whether such a table is, in fact, needed can be best made by the process which calls for the table.

In one embodiment of the invention, when a call is made for a required
 5 table, the font subsystem operates in a first mode where it automatically issues a command to synthesize the table, if it is not already present. In this case, the absence of the data table from the original font is transparent to the calling process. If the requested table is an optional table, however, the font subsystem does not automatically initiate the synthesis of the table. Rather, it operates in a
 10 second mode where it returns a message to the calling process, e.g. the layout processor, which indicates that the table is not available. At this point, the processor can determine whether the table is really needed. If so, it issues an explicit call to the font subsystem to create the table. In response to this call, the font subsystem then initiates the synthesis of the table.

As a further feature of the invention, once a table is synthesized for a
 15 particular font, the table is stored in a persistent manner, so that its data is available for all subsequent sessions which employ that font. Preferably, the synthesized table is not stored as part of the original font definition, i.e. it is not contained within the suitcase 46 for the font. Rather, as depicted in Figure 5, the
 20 table is stored in a separate annex 50 that is associated with the font. This annex is a file which contains additional information about the font, namely synthesized tables and the like, but does not affect the original font definition stored in the suitcase. By storing the synthesized tables in a separate annex, font protection techniques which verify the integrity of the original font data in the suitcase 46 will
 25 continue to operate in the intended manner. For example, some font protection techniques rely upon a checksum of data in the font, and/or a digital signature of the font. These techniques need not consider, nor even be aware of, the annex file when computing the checksums or digital signatures. Hence, the values which they compute will continue to indicate that the original font data is unmodified.

The table data can be stored in the annex in one of two ways. If all tables are synthesized at once, it may be preferable to store the tables in the same type of data structure as the original font data. For instance, this structure may include a directory, and the tables are stored with offsets and lengths that are specified in the directory. In this manner, data in the annex file can be indexed in the same manner as data in the original font file.

If the tables are synthesized one at a time, rather than all at once, it may be preferable to store each table individually, e.g. as a linked list of tables that form the annex file. In this situation, the overhead of a directory can be eliminated.

The general operation of the line layout processor and font subsystem, in accordance with the present invention, is depicted in the flowchart of Figure 6. When a particular line layout procedure is to be performed, e.g. metamorphosis, the layout processor issues a call at step 60 for the font tables that are pertinent to that procedure. At step 62, a determination is made by the font subsystem whether the requested tables are present. If so, they are returned to the layout processor and the appropriate procedure is carried out at step 64. The processor then goes to the next procedure to be performed, at step 66, or terminates at step 68 if no further procedures are to be carried out.

If the font subsystem determines at step 62 that the font suitcase does not contain the requested table, it then looks to see whether an annex is associated with the font, at step 70. If so, the font subsystem attempts to obtain the table from the annex at step 72. If the font subsystem obtains the table, at step 74, it returns it to the processor, which proceeds in the normal fashion, at step 64, to carry out the current procedure.

If there is no annex associated with the font, as determined at step 70, then the requested data table is not present. Therefore, the table synthesis routine is initiated at step 76. As described previously, this routine can be automatically called by the font subsystem if the data table is classified as being required, or in response to a call from another process if the table is an optional one. Once the appropriate table has been synthesized, the annex is created at step 78 and the table

is stored in it, and then used to complete the procedure at step 64. If the annex was already present, but the requested data table was not found at step 74, the table is synthesized at step 79 and then added to the annex at step 80, where it is used to carry out the current procedure at step 64.

- 5 The process that is carried out at steps 76 and 80 to synthesize certain types of data tables is schematically depicted in Figure 7. In general, the process is carried out by a synthesizer which contains tables of information that are generic to all fonts for which synthesis can be carried out. The information in these tables is combined with font-specific information to synthesize the necessary data table.
- 10 The synthesizer also contains a program to assemble data from various sources into the appropriate table.

- The first step of the process is to build a font map 81. The map is constructed from the repertoire of data that is available for each of the glyphs in a specific font for which a data table is to be synthesized. In essence, the glyph
- 15 repertoire consists of all of the information that is contained within a font, e.g. in the font suitcase, about individual glyphs. In a TrueType font, for example, the pertinent data might consist of a 'cmap' table which identifies the mappings between character encodings and glyph indices, a 'post' table which provides the PostScript names for all of the glyphs in a font, as well as other PostScript data.
- 20 In some cases, additional information that is known about a font may be available, even though it is not found in the font suitcase. This data can be stored in a table that is contained within the synthesizer.

- The font map which is constructed from the glyph repertoire provides certain characterizing information about each glyph in the font. In the example of
- 25 Figure 7, the data for each glyph includes a glyph number, an identification value, properties, and a unicode value. The glyph number is specific to the particular font, and can be obtained from data tables in the font suitcase 46. For instance, Figure 7 illustrates a table 47 which maps each glyph to an associated name and character encoding. In contrast, the identification value is universal to all fonts.
- 30 For instance, glyph 31 in font 1 may have the same appearance as glyph 42 in font

2. These two glyphs are represented by the same universal identification number, e.g. 1300 in the example of Figure 7. These universal identification numbers are stored in a table 82, which forms part of the synthesizer. In the illustrated example, the name of a glyph in the font-specific table 47 is used to locate its universal ID in the table 82.

The properties value indicates characteristics of the glyph. For instance, the properties associated with the glyph 'A' are that it is a letter, and it has left-to-right directionality. The unicode value is a character encoding value which is independent of the computer platform. These entries are obtained from the table 82 for storage in the font map 81.

In the example of Figure 7, the synthesizer is building a data table which contains a mapping from uppercase to lowercase letters. This process utilizes a mapping table 84 in the synthesizer, which defines relationships between characters. The mapping table 84 is based upon universal identification values, whereas the synthesized table must identify the glyph values for the specific font of interest in order to be useable by the layout processor. For the first entry in the font map, associated with the uppercase character 'A', the table 84 indicates that the glyph with the identification value 1300 maps to glyph ID 11005. The synthesizer then looks in the font map for any glyph having the identification value 11005. In the example of Figure 7, glyph 46, corresponding to the lowercase letter 'a', meets this criterion. Accordingly, an entry is made in the table 86 being created, which indicates that glyph 31 maps to glyph 46. A corresponding entry is created for each glyph in the font map whose identification value has an associated uppercase-to-lowercase mapping in the synthesizer table 84. Once all of these mappings have been identified and entered in the synthesized data table 86, the table is stored in the annex.

In a similar manner, other types of metamorphosis tables can be created. For instance, a ligature table which maps a sequence of consecutive glyphs to a single ligature glyph can be created using the same procedure. These mappings are stored in another mapping table which forms part of the synthesizer. The data

in the table comprises font-specific information that is derived from font-independent data stored in the synthesizer.

The synthesis of tables such as uppercase-to-lowercase and ligature tables, as described in the foregoing example, is based upon data which is internally stored in the synthesizer. The synthesis of other types of tables can be carried out on the basis of information which is obtained externally of the synthesizer. For instance, a TrueType font contains three primary types of resources, known as sfnt, NFNT and FOND, as depicted in Figure 8. The sfnt resource forms the font definition that is used by the layout processor. If a required data table is not present in the sfnt resource, the layout processor must obtain it from the annex file.

In some cases, the information necessary to perform a layout operation may be present in one of the other resources of the suitcase. One example relates to kerning data. It may be the case that the sfnt resource does not contain the required kerning table, in which case the layout processor is not able to utilize data directly from the suitcase to perform an operation. However, kerning data may be present in the FOND resource, although not in a format which can be employed by the layout processor. In this case, the synthesizer can use this data in the FOND resource to build the necessary table. In operation, if a call is made to synthesize a kerning table, the synthesizer examines the FOND resource, as well as any other resource in the font suitcase, to determine whether kerning data is present. If so, the data is retrieved and stored in a data structure having the format that can be read by the layout processor. This data structure is stored in the annex file as the data table, where it can be used by the layout processor. In this embodiment, therefore, the data table is not computed from a variety of different data sources, as in the example of Figure 7. Rather, the synthesis of the data table involves the translation, or reformatting, of data that already exists in the font file.

The building of a new table from data which exists in the font can be performed for a variety of purposes. For instance, the data needed to perform a particular operation may exist in a number of different tables. To optimize the

procedure, a new table can be constructed in which the data is collected into a single data structure, where it can be retrieved more efficiently by the process. As a further example, it may be desirable to construct a new table to correct an error in the original data, e.g. eliminate references to corrupted data.

5 From the foregoing, it can be seen that the present invention provides a mechanism via which data tables pertaining to characteristics of fonts can be automatically synthesized when they are missing from an original font definition, to thereby enable newly developed font processing technologies to be employed in connection with fonts that were created before such technologies became available.

10 By storing the synthesized data tables in an annex associated with the font, the original font definition is not modified. As a result, font protection mechanisms continue to operate as designed. Furthermore, the annex file remains persistent across successive boots of the computer, so that the effort required to synthesize a table needs to be expended only one time per font.

15 It will be appreciated by those of ordinary skill in the art that the present invention can be embodied in other forms without departing from the spirit or essential characteristics thereof. For instance, although the synthesis of data tables has been specifically described in connection with the operation of the layout processor, such a procedure can be employed by any component of a font system

20 which makes use of font data tables. The presently disclosed embodiments are therefore considered in all respects to be illustrative, and not restrictive. The scope of the invention is indicated by the appended claims, rather than the foregoing description, and all changes that come within the meaning and range of equivalence thereof are intended to be embraced therein.

What Is Claimed Is:

1. A method for generating an image of a sequence of characters, comprising the steps of:

5 retrieving glyphs from a font which correspond to characters in a string of characters;

determining whether the font contains a predetermined data table that pertains to the layout of glyphs;

10 automatically synthesizing said data table, based upon data contained in the font, if the font is determined not to contain said data table; laying out the glyphs in a line, in accordance with the data in said table; and

generating an image of the laid-out line of glyphs.

2. The method of claim 1 wherein said generating step includes displaying the line of glyphs on a display device.

15 3. The method of claim 1 wherein said generating step includes printing the line of glyphs in a document.

4. The method of claim 1 further including the step of storing the synthesized data table in a persistent annex file that is associated with, but separate from, the font.

20 5. The method of claim 1 further including the step of determining whether said data table is stored in an annex file associated with the font, and wherein said automatic synthesis step is carried out only if the table is not contained in either the font or the annex file.

6. The method of claim 1 wherein the step of automatically synthesizing said data table comprises the steps of:

building a font map that contains information about individual glyphs in the font;

5 determining relationships between items of information in the font map; and

constructing a table which identifies said relationships.

7. The method of claim 6 wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

10 8. The method of claim 7 wherein the synthesized table contains font-specific information that is determined with reference to generic information.

9. The method of claim 1 wherein the step of automatically synthesizing said data table comprises the steps of retrieving data from the font and storing the retrieved data in a table having a predetermined data format.

15 10. The method of claim 1 further including the steps of determining whether said data table is of a first type or a second type when the data table is determined not to be present in the font; directly initiating said synthesizing step if said data table is of said first type; or, providing an indication that said data table is not present in the font if said data table is of said second type, and initiating said
20 synthesizing step upon receipt of a request that is responsive to said indication.

11. A system for generating images of characters, comprising:
a font subsystem which is responsive to identification of characters
to access at least one font file to retrieve glyphs associated with the identified
characters, and data tables that contain information about glyphs in the font; and
5 a font table synthesizer which is responsive to the absence of a
predetermined data table for creating and storing said table on the basis of data
contained in the font file.
12. The system of claim 11 wherein said font subsystem determines
whether a predetermined data table is contained in the font file, and causes said
10 synthesizer to create said table when a determination is made that the table is not
present in the font file.
13. The system of claim 11 wherein said font synthesizer stores said
table in an annex file that is associated with, but separate from, the font file.
14. The system of claim 13 wherein said font subsystem determines
15 whether a predetermined data table is contained in either the font file or the annex
file, and causes said synthesizer to create said table when a determination is made
that the table is not present in either the font file or the annex file.
15. The system of claim 12 wherein said font subsystem operates in a
first mode to cause said synthesizer to automatically create the table in response to
20 said determination, and in a second mode to provide an indication when a data
table is determined not to be present and thereafter cause said synthesizer to create
the table in response to a request that is responsive to said indication.

16. A method for automatically synthesizing a data table that contains information about glyphs in a font, comprising the steps of:

building a font map that contains information about individual glyphs in the font;

5 determining relationships between items of information in the font map; and

constructing a table which identifies said relationships.

17. The method of claim 16 wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

10 18. The method of claim 17 wherein the synthesized table contains font-specific information that is determined with reference to generic information.

19. A method for providing data that relates to the implementation of a font, comprising the steps of:

15 receiving a request for a data table that pertains to the implementation of a font;

determining whether the data table is present in a file containing the font; and

synthesizing said table from data contained in said file if the table is not present in the font file.

20 20. The method of claim 19 further including the step of storing the synthesized table in an annex file separate from said font file.

21. The method of claim 19 further including the steps of determining whether said data table is of a first type or a second type when the data table is determined not to be present in the font file; automatically initiating said synthesizing step if said data table is of said first type; or, providing an indication
5 that said data table is not present in the font file if said data table is of said second type, and initiating said synthesizing step upon receipt of a request that is responsive to said indication.

22. The method of claim 19 wherein the step of synthesizing said data table comprises the steps of:
10 building a font map that contains information about individual glyphs in the font;
determining relationships between items of information in the font map; and
constructing a table which identifies said relationships.

23. The method of claim 22 wherein some of the information in said
15 font map is specific to the font, and other information is generic to multiple fonts.

24. The method of claim 23 wherein the synthesized table contains font-specific information that is determined with reference to generic information.

25. The method of claim 19 wherein the step of synthesizing said data
20 table comprises the steps of retrieving data from the font and storing the retrieved data in a table having a predetermined data format.

26. A computer-readable medium containing a program which executes the steps of:

receiving a request for a data table that pertains to the implementation of a font;

5 determining whether the data table is present in a file containing the font; and

synthesizing said table from data contained in said file if the table is not present in the font file.

27. The computer-readable medium of claim 26, wherein said program
10 executes the further step of storing the synthesized table in an annex file separate from said font file.

28. The computer-readable medium of claim 26, wherein said program
executes the further step of determining whether said data table is of a first type or
a second type when the data table is determined not to be present in the font file;
15 automatically initiating said synthesizing step if said data table is of said first type;
or, providing an indication that said data table is not present in the font file if said
data table is of said second type, and initiating said synthesizing step upon receipt
of a request that is responsive to said indication.

29. A computer-readable medium containing a program which executes
20 the steps of:

building a font map that contains information about individual glyphs in a font;

determining relationships between items of information in the font map; and

25 constructing a table which identifies said relationships.

30. The computer-readable medium of claim 29, wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

31. The computer-readable medium of claim 30, wherein the
5 synthesized table contains font-specific information that is determined with
reference to generic information.

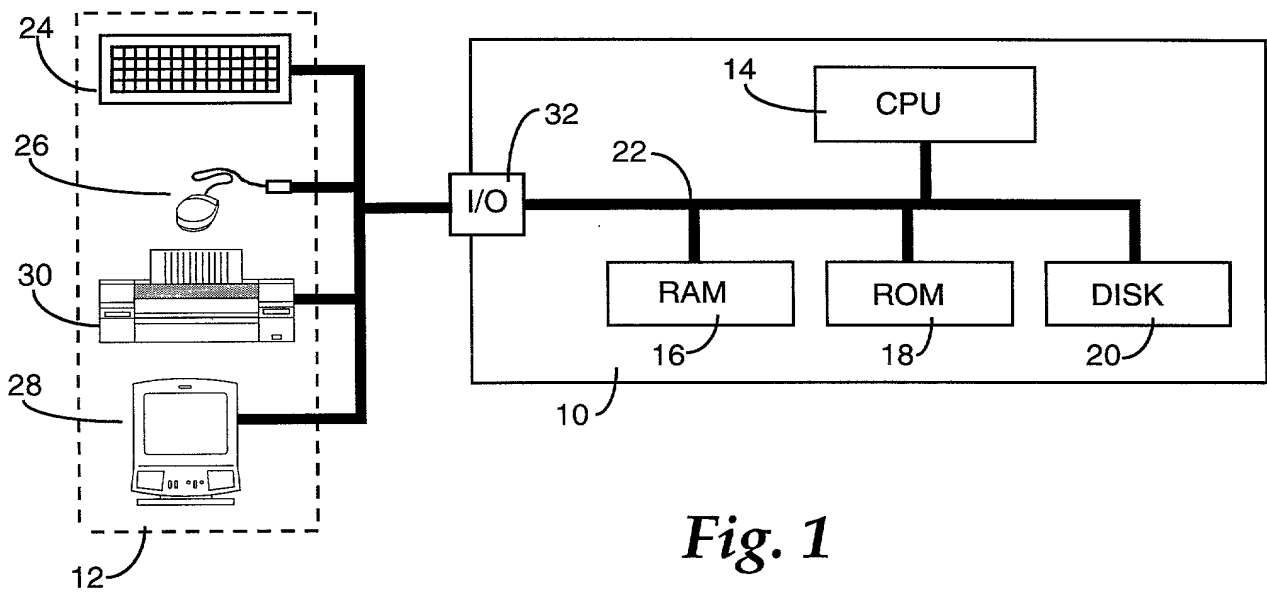


Fig. 1

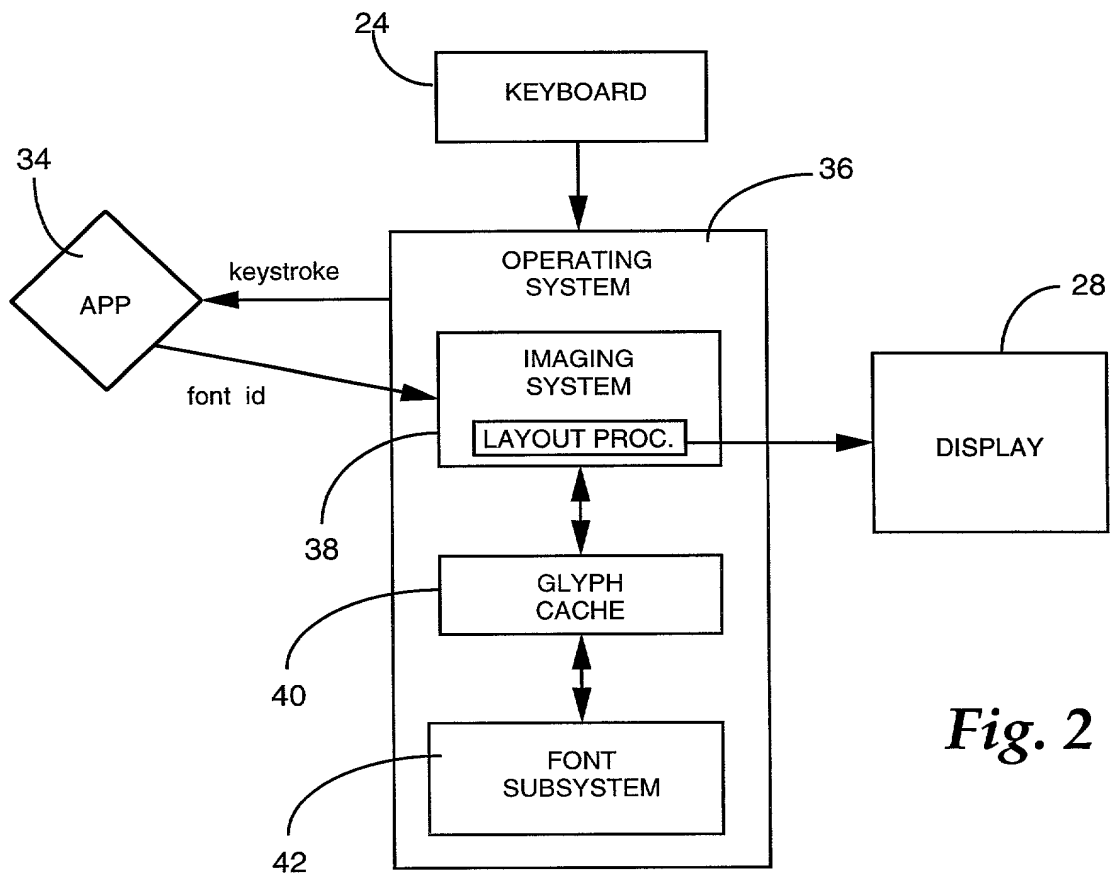


Fig. 2

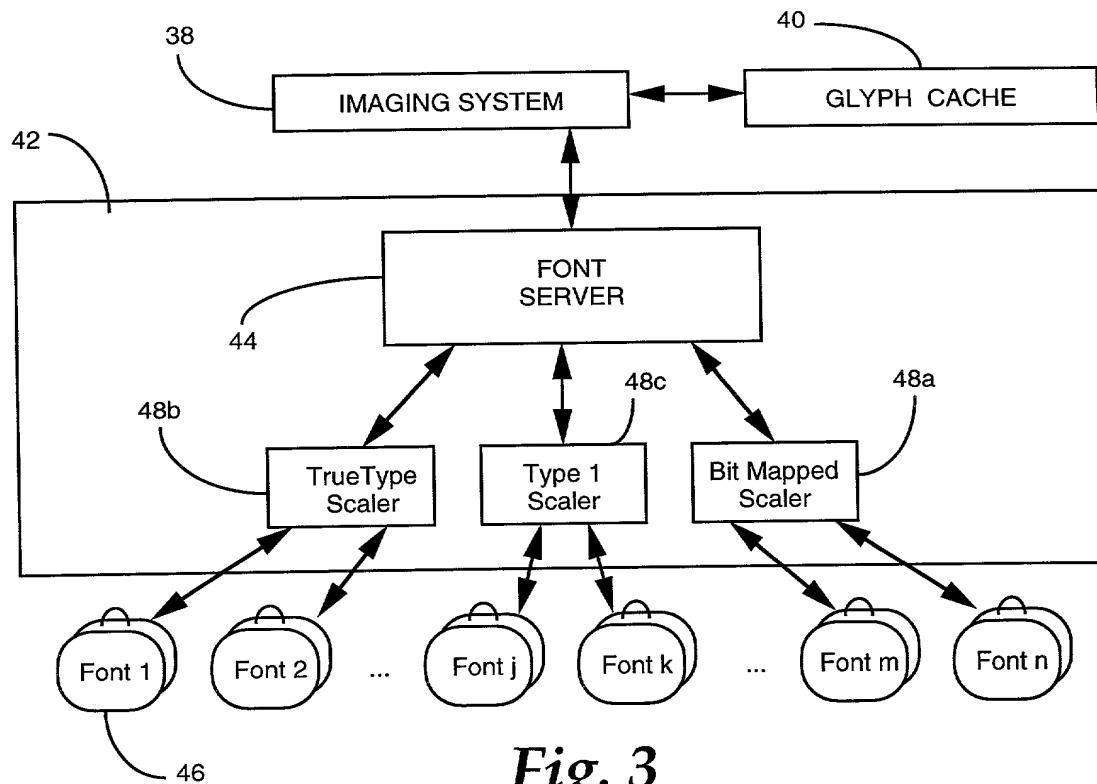


Fig. 3

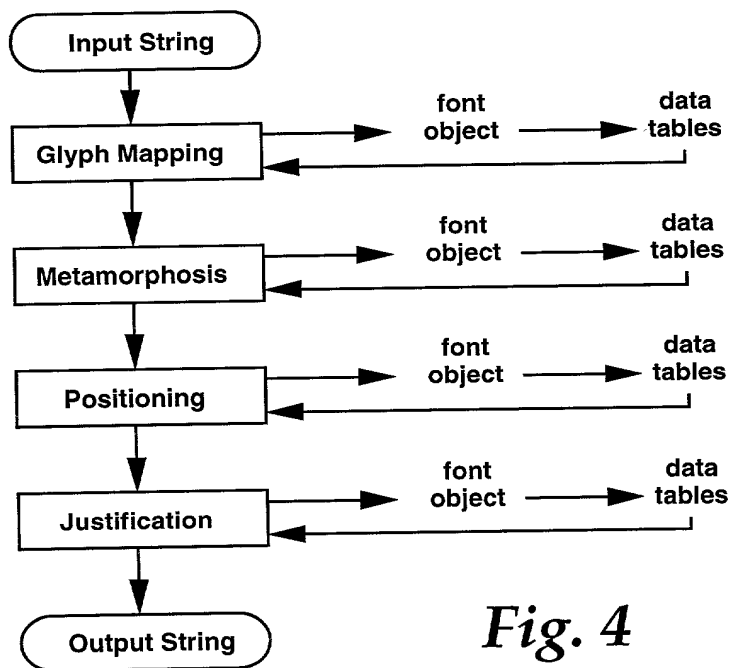


Fig. 4

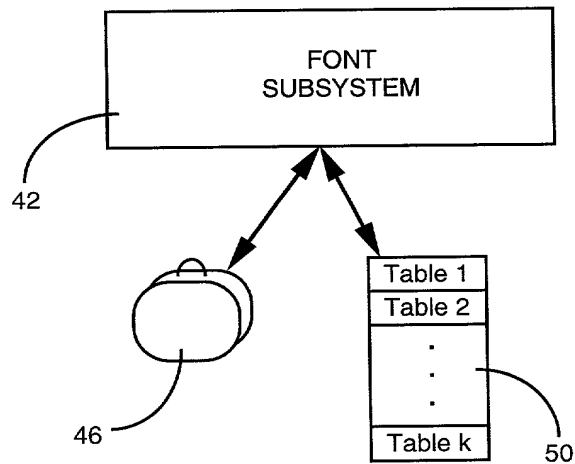


Fig. 5

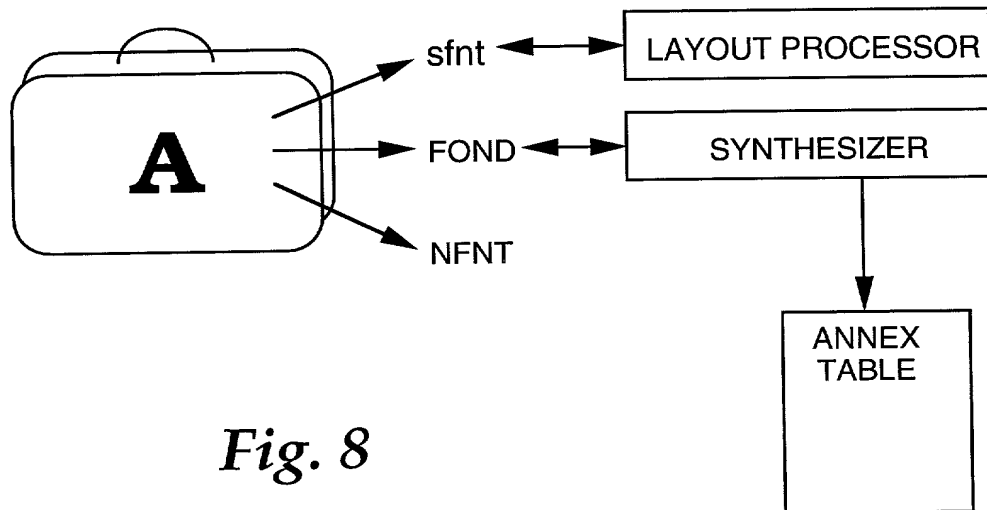


Fig. 8

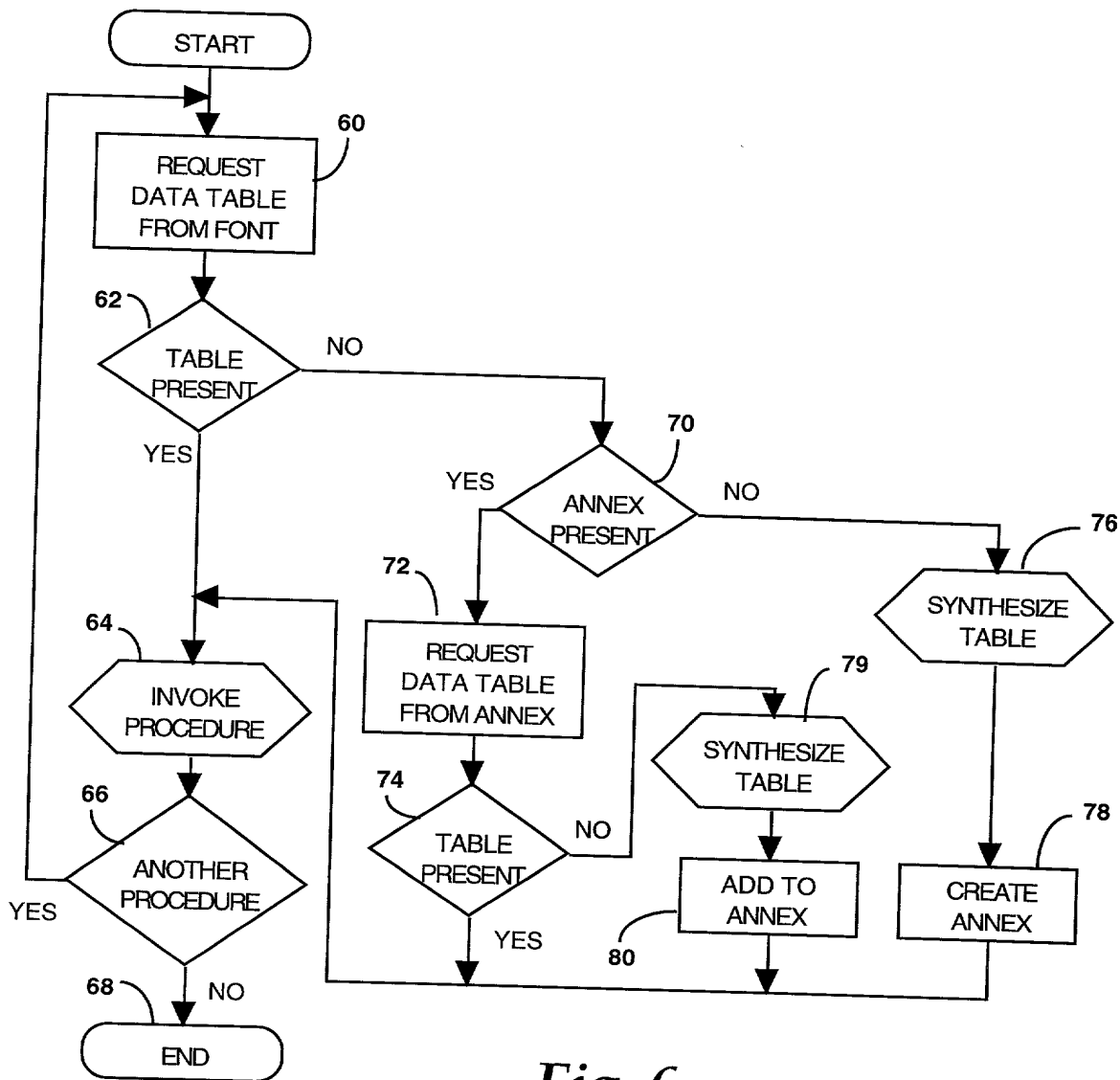


Fig. 6

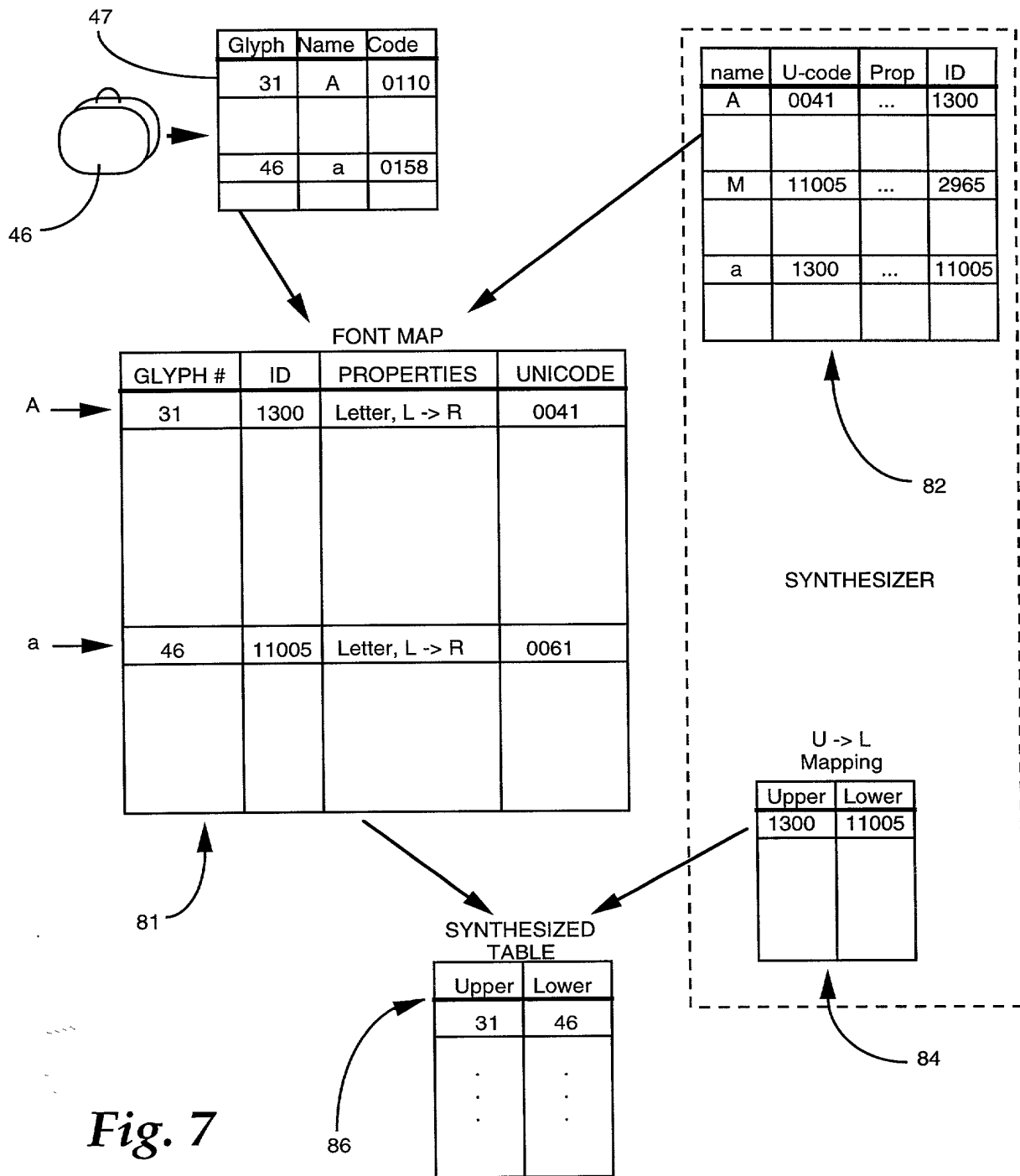


Fig. 7

COMBINED DECLARATION AND POWER OF ATTORNEY FOR UTILITY PATENT APPLICATION

Attorney's Docket No.

P2380-505

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I BELIEVE I AM THE ORIGINAL, FIRST AND SOLE INVENTOR (if only one name is listed below) OR AN ORIGINAL, FIRST AND JOINT INVENTOR (if more than one name is listed below) OF THE SUBJECT MATTER WHICH IS CLAIMED AND FOR WHICH A PATENT IS SOUGHT ON THE INVENTION ENTITLED:

AUTOMATIC SYNTHESIS OF FONT TABLES FOR CHARACTER LAYOUT

the specification of which

(check one)

☒ is attached hereto;

☐ was filed on _____ as

Application No. _____

and was amended on _____;
(if applicable)

I HAVE REVIEWED AND UNDERSTAND THE CONTENTS OF THE ABOVE-IDENTIFIED SPECIFICATION, INCLUDING THE CLAIMS, AS AMENDED BY ANY AMENDMENT REFERRED TO ABOVE;

I ACKNOWLEDGE THE DUTY TO DISCLOSE TO THE OFFICE ALL INFORMATION KNOWN TO ME TO BE MATERIAL TO PATENTABILITY AS DEFINED IN TITLE 37, CODE OF FEDERAL REGULATIONS, Sec. 1.56 (as amended effective March 16, 1992);

I do not know and do not believe the said invention was ever known or used in the United States of America before my or our invention thereof, or patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to said application; that said invention was not in public use or on sale in the United States of America more than one year prior to said application; that said invention has not been patented or made the subject of an inventor's certificate issued before the date of said application in any country foreign to the United States of America on any application filed by me or my legal representatives or assigns more than twelve months prior to said application;

I hereby claim foreign priority benefits under Title 35, United States Code Sec. 119 and/or Sec. 365 of any foreign application(s) for patent or inventor's certificate as indicated below and have also identified below any foreign application for patent or inventor's certificate on this invention having a filing date before that of the application(s) on which priority is claimed:

COMBINED DECLARATION AND POWER OF ATTORNEY			Attorney's Docket No. P2380-505																																																																																					
COUNTRY/INTERNATIONAL	APPLICATION NUMBER	DATE OF FILING (day, month, year)	PRIORITY CLAIMED																																																																																					
			YES NO																																																																																					
			YES NO																																																																																					
<p>I hereby appoint the following attorneys and agent(s) to prosecute said application and to transact all business in the Patent and Trademark Office connected therewith and to file, prosecute and to transact all business in connection with international applications directed to said invention:</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 33%;">William L. Mathis</td> <td style="width: 10%;">17,337</td> <td style="width: 33%;">James A. LaBarre</td> <td style="width: 10%;">28,632</td> <td style="width: 33%;">Matthew L. Schneider</td> <td style="width: 10%;">32,814</td> </tr> <tr> <td>Peter H. Smolka</td> <td>15,913</td> <td>E. Joseph Gess</td> <td>28,510</td> <td>Michael G. Savage</td> <td>32,596</td> </tr> <tr> <td>Robert S. Swecker</td> <td>19,885</td> <td>R. Danny Huntington</td> <td>27,903</td> <td>Gerald F. Swiss</td> <td>30,113</td> </tr> <tr> <td>Platon N. Mandros</td> <td>22,124</td> <td>Eric H. Weisblatt</td> <td>30,505</td> <td>Michael J. Ure</td> <td>33,089</td> </tr> <tr> <td>Benton S. Duffett, Jr.</td> <td>22,030</td> <td>James W. Peterson</td> <td>26,057</td> <td>Charles F. Wieland III</td> <td>33,096</td> </tr> <tr> <td>Norman H. Stepno</td> <td>22,716</td> <td>Teresa Stanek Rea</td> <td>30,427</td> <td>Bruce T. Wieders</td> <td>33,815</td> </tr> <tr> <td>Ronald L. Grudziecki</td> <td>24,970</td> <td>Robert E. Krebs</td> <td>25,885</td> <td>Todd R. Walters</td> <td>34,040</td> </tr> <tr> <td>Frederick G. Michaud, Jr.</td> <td>26,003</td> <td>William C. Rowland</td> <td>30,888</td> <td>Ronni S. Jilkons</td> <td>31,979</td> </tr> <tr> <td>Alan E. Kopecki</td> <td>25,813</td> <td>T. Gene Dillamnty</td> <td>25,423</td> <td>Harold R. Brown III</td> <td>36,341</td> </tr> <tr> <td>Regis E. Slutter</td> <td>26,999</td> <td>Patrick C. Keane</td> <td>32,858</td> <td>Allen R. Baum</td> <td>36,086</td> </tr> <tr> <td>Sammel C. Miller III</td> <td>27,360</td> <td>Bruce J. Boggs, Jr.</td> <td>32,344</td> <td>Steven M. du Bois</td> <td>35,023</td> </tr> <tr> <td>Ralph L. Freeland, Jr.</td> <td>16,110</td> <td>William H. Benz</td> <td>25,952</td> <td>Brian P. O'Shaughnessy</td> <td>32,747</td> </tr> <tr> <td>Robert G. Mukai</td> <td>28,531</td> <td>Peter K. Skiff</td> <td>31,917</td> <td> </td> <td> </td> </tr> <tr> <td>George A. Hovanec, Jr.</td> <td>28,223</td> <td>Richard J. McGrath</td> <td>29,195</td> <td> </td> <td> </td> </tr> </table>					William L. Mathis	17,337	James A. LaBarre	28,632	Matthew L. Schneider	32,814	Peter H. Smolka	15,913	E. Joseph Gess	28,510	Michael G. Savage	32,596	Robert S. Swecker	19,885	R. Danny Huntington	27,903	Gerald F. Swiss	30,113	Platon N. Mandros	22,124	Eric H. Weisblatt	30,505	Michael J. Ure	33,089	Benton S. Duffett, Jr.	22,030	James W. Peterson	26,057	Charles F. Wieland III	33,096	Norman H. Stepno	22,716	Teresa Stanek Rea	30,427	Bruce T. Wieders	33,815	Ronald L. Grudziecki	24,970	Robert E. Krebs	25,885	Todd R. Walters	34,040	Frederick G. Michaud, Jr.	26,003	William C. Rowland	30,888	Ronni S. Jilkons	31,979	Alan E. Kopecki	25,813	T. Gene Dillamnty	25,423	Harold R. Brown III	36,341	Regis E. Slutter	26,999	Patrick C. Keane	32,858	Allen R. Baum	36,086	Sammel C. Miller III	27,360	Bruce J. Boggs, Jr.	32,344	Steven M. du Bois	35,023	Ralph L. Freeland, Jr.	16,110	William H. Benz	25,952	Brian P. O'Shaughnessy	32,747	Robert G. Mukai	28,531	Peter K. Skiff	31,917			George A. Hovanec, Jr.	28,223	Richard J. McGrath	29,195		
William L. Mathis	17,337	James A. LaBarre	28,632	Matthew L. Schneider	32,814																																																																																			
Peter H. Smolka	15,913	E. Joseph Gess	28,510	Michael G. Savage	32,596																																																																																			
Robert S. Swecker	19,885	R. Danny Huntington	27,903	Gerald F. Swiss	30,113																																																																																			
Platon N. Mandros	22,124	Eric H. Weisblatt	30,505	Michael J. Ure	33,089																																																																																			
Benton S. Duffett, Jr.	22,030	James W. Peterson	26,057	Charles F. Wieland III	33,096																																																																																			
Norman H. Stepno	22,716	Teresa Stanek Rea	30,427	Bruce T. Wieders	33,815																																																																																			
Ronald L. Grudziecki	24,970	Robert E. Krebs	25,885	Todd R. Walters	34,040																																																																																			
Frederick G. Michaud, Jr.	26,003	William C. Rowland	30,888	Ronni S. Jilkons	31,979																																																																																			
Alan E. Kopecki	25,813	T. Gene Dillamnty	25,423	Harold R. Brown III	36,341																																																																																			
Regis E. Slutter	26,999	Patrick C. Keane	32,858	Allen R. Baum	36,086																																																																																			
Sammel C. Miller III	27,360	Bruce J. Boggs, Jr.	32,344	Steven M. du Bois	35,023																																																																																			
Ralph L. Freeland, Jr.	16,110	William H. Benz	25,952	Brian P. O'Shaughnessy	32,747																																																																																			
Robert G. Mukai	28,531	Peter K. Skiff	31,917																																																																																					
George A. Hovanec, Jr.	28,223	Richard J. McGrath	29,195																																																																																					
<p>and: _____</p>																																																																																								
<p>Address all correspondence to: <u>James W. Peterson</u> <u>BURNS, DOANE, SWECKER & MATHIS, L.L.P.</u> <u>P.O. Box 1404</u> <u>Alexandria, Virginia 22313-1404</u></p>																																																																																								
<p>Address all telephone calls to: <u>James A. LaBarre</u> at (703) 836-6620.</p>																																																																																								
<p>I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.</p>																																																																																								
FULL NAME OF SOLE OR FIRST INVENTOR		SIGNATURE	DATE																																																																																					
David G. Opatad		<i>David G. Opatad</i>	7 May 1999																																																																																					
RESIDENCE		CITIZENSHIP																																																																																						
450 Sierra Vista Avenue, #4, Mountain View, California 94043, UNITED STATES OF AMERICA		United States of America																																																																																						
POST OFFICE ADDRESS																																																																																								
450 Sierra Vista Avenue, #4, Mountain View, California 94043, UNITED STATES OF AMERICA																																																																																								
FULL NAME OF SECOND JOINT INVENTOR, IF ANY		SIGNATURE	DATE																																																																																					
Alexander B. Beamman		<i>Alexander B. Beamman</i>	5/7/99																																																																																					
RESIDENCE		CITIZENSHIP																																																																																						
4024 Ross Park Court, San Jose, California 95118, UNITED STATES OF AMERICA		United States of America																																																																																						
POST OFFICE ADDRESS																																																																																								
4024 Ross Park Court, San Jose, California 95118, UNITED STATES OF AMERICA																																																																																								
FULL NAME OF THIRD JOINT INVENTOR, IF ANY		SIGNATURE	DATE																																																																																					
RESIDENCE		CITIZENSHIP																																																																																						
POST OFFICE ADDRESS																																																																																								